(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 080 266**
A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 82305618.9

(22) Date of filing: 21.10.82

(51) Int. Cl.³: G 06 F 15/332

(30) Priority: 24.11.81 GB 8135328

(43) Date of publication of application: 01.06.83 Bulletin 83/22

(84) Designated Contracting States: DE FR IT NL

(71) Applicant: ITT INDUSTRIES INC., 320 Park Avenue, New York, NY 10022 (US)
(84) Designated Contracting States: FR IT NL

(71) Applicant: Deutsche ITT Industries GmbH, Hans-Bunte-Strasse 19 Postfach 840, D-7800 Freiburg (DE)
(84) Designated Contracting States: DE

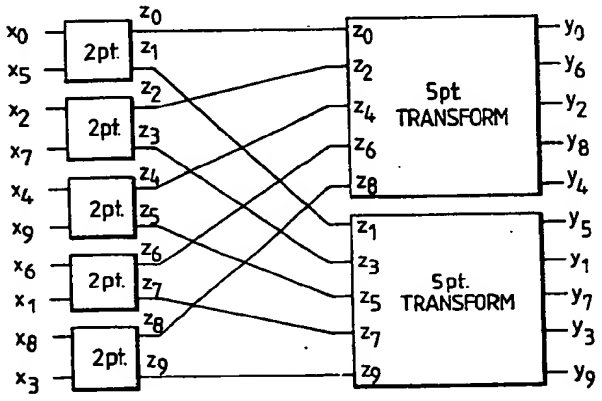
(72) Inventor: Rossiter, Timothy John Miles, 10 Yewlands, Sawbridgeworth CM21 9NP Hertfordshire (GB)

(74) Representative: Capsey, Sydney Ross et al, ITT UK Patent Department Maidstone Road, Foots Cray Sidcup DA14 5HT (GB)

(54) Discrete fourier transform circuit.

(57) In a transmultiplexer, which is an interface between a number of PCM highways and an FDM group, the conversion between PCM and FDM involves first converting the PCM words into their linear form and filtering them, applying th m to a circuit which performs a discrete Fourier transf rm (DFT) on them followed by a second stage of filtering. An efficient implementation of a DFT is described which reduces the number of multiplications required by making use of certain symmetry properties in a DFT matrix.

The whole can be used «backwards», as it were to give an inverse Fourier transform as needed for the FDM–PCM conversion.



EP 0 080 266 A2

0080266

T.J.M. ROSSITER-4

DISCRETE FOURIER TRANSFORM CIRCUIT

This invention relates to an electrical circuit arrangement for implementing a discrete Fourier transform (DFT), and also for implementing an inverse Fourier transform(IFT).

Such circuits are usable in digital signal processing applications and in one application a circuit comprising digital filters and a DFT implementation is used to provide the equivalent of a set of filters used to assemble a frequency division multiplex (FDM) group from a number of PCM channels after the latter have been converted into a linear form. After the signals have been passed through the digital filtering arrangement thus formed they are passed to the FDM output via digital to analogue conversion circuitry.

An object of the invention is to provide an economical circuit for implementing a DFT.

Thus we provide an electronic circuit for the implementation of a discrete Fourier transform (DFT), which includes input means to which are applied electrical signals representing quantities on which the DFT is to be performed, and a sum and difference processor (SDP) to which those signals are applied and which can perform calculations of the following types on the signals:-

$$u_o = x_o$$

$$u_k = x_k + x_N{}_{-k}$$

$$v_k = x_k - x_N{}_{-k}$$

where x_2 are a sequence of input samples and N is the order of the matrix, x is an input signal and U and V are the results of the computations. According to the invention, in an arrangement as set out above, the results generated by the SDP are collected in an addressable buffer store, and those results as assembled in the addressable buffer store are applied to a sum of products multiplier (SPM) which evaluates expressions in the following format, quoted for an example in which N = 5;

$$Yr_0 = Tr_0$$
$$Yi_0 = Ti_0$$
$$Yr_1 = Ur_0 + a_1 Ur_1 - b_1 Vi_1 + a_2 Ur_2 - b_2 Vi_2$$
$$Yi_1 = Ui_0 + a_1 Ui_1 + b_1 Vr_1 + a_2 Ui_2 + b_2 Vr_2$$
$$Yr_2 = Ur_0 + a_1 Ur_2 + b_1 vi_2 + a_2 Ur_1 - b_2 Vi_1$$
$$Yi_2 = Ui_0 + a_1 Ui_2 - b_1 Vr_2 + a_2 Ui_1 + b_2 Vr_1$$
$$Yr_3 = Ur_0 + a_1 Ur_2 - b_1 Vi_2 + a_2 Ur_1 + b_2 Vi_1$$
$$Yi_3 = Ui_0 + a_1 Ui_2 + b_1 Vr_2 + a_2 Ui_1 - b_2 Vr_1$$
$$Yr_4 = Ur_0 + a_1 Ur_1 + b_1 Vi_1 + a_2 Ur_2 + b_2 Vi_2$$
$$Yi_4 = Ui_0 + a_1 Ui_1 - b_1 Vr_1 + a_2 Ui_2 - b_2 Vr_2$$

where Yr and Yi are real and imaginary results respectively of the DFT, where a_1, b_1, a_2, b_2 are the real and imaginary Fourier coefficients, and where

$$a_1 + jb_1 = \exp\left(j\frac{2}{5}\right), \quad a_2 + jb_2 = \exp\left(j\frac{2}{5}\right)$$

$$Tr_0 = Ur_0 + Ur_1 + Ur_2$$
$$Ti_0 = Ui_0 + Ui_1 + Ui_2,$$

the resulting output signals from the SPM being representative of a DFT of the quantities represented by the electrical signals applied to said input means.

A transmultiplexer is a circuit which acts as a bidirectional interface between a set of PCM highways and a collection of FDM channels. Thus with such a circuit it is also necessary to be able to convert from the FDM format into the PCM format, and in accordance with the invention this is done by the

use of an arrangement which includes filters and an inverse DFT (IDFT). The IDFT is implemented in a very similar manner to the DFT, except that the values at coefficients are different and the sequencing or certain data is changed.

Thus we have a filter arrangement which in effect uses the DFT circuit in the reverse sense to that used for the PCM to FDM transformation. The arrangements to be described herein are intended for use in a transmultiplexer which interfaces between two 30-channel PCM systems and five 12-channel FDM groups.

Embodiments of the invention will now be described with reference to the accompanying drawings in which:

Fig. 1 represents schematically a 10 point transform factorised into 2 point and 5 point transforms.

Fig. 2 represents schematically a 12 point transform factorised into 4 point and 3 point transforms.

Fig. 3 shows a general scheme for implementing an efficient Fourier transform.

Fig. 4 represents data formats used in the DFT evaluation.

Fig. 5 is an outline representation of an SDP.

Fig. 6 is an addressable buffer store (ABS) as used in the arrangement of Fig. 3.

First we consider the theory of the DFT, and show how a number of novel developments lead to an implementation which requires substantially less multiplications than a straightforward DFT implementation. As such it can be called an FFT. We start with a mathematical derivation of the algorithm, and follow this with an implementation thereof, suitable for implementation using LSI digital circuits.

The Discrete Fourier Transform (DFT)

The DFT can be expressed as follows:-

$$y_p = \sum_{q=0}^{N-1} x_q . w^{pq} \qquad p,q,E[1,N-1] \qquad (1)$$

where $w = N_1$ or alternatively $w = \exp\left(j \frac{2}{N}\right.$ (2)

("j" denotes the square root of -1).

Equation (1) can be expressed in matrix form, as given where N = 5 in equation (3).

$$\begin{matrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{matrix} \quad \begin{matrix} w^0 & w^0 & w^0 & w^0 & w^0 \\ w^0 & w^1 & w^2 & w^3 & w^4 \\ w^0 & w^2 & w^4 & w^1 & w^3 \\ w^0 & w^3 & w^1 & w^4 & w^2 \\ w^0 & w^4 & w^3 & w^2 & w^1 \end{matrix} \quad \begin{matrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{matrix}$$

The left-hand side of this is a column vector of output samples $[y_p]$, and the right-hand side is the square DFT matrix (referred to as the W matrix) operating on a column vector of input samples $[x_q]$.

The elements of the DFT or W matrix can be derived as follows. The value of the element in the r'th row and p'th column is obtained by evaluating expression (4).

$$w^{(p.q \bmod N)} \qquad (4)$$

The power of W can be evaluated modulo N since:

$$w^p = w^{p+N} \qquad (5)$$

In equation (3) and the examples discussed the number of points in the data set operated upon (i.e. the value of N) is taken to be five. This does not imply any restriction on the value of N, and in fact different values are used in the transmultiplexer application referred to. The basic DFT (as given in (1) or (3) holds for all values of N.

Equation (3) is the basis for the efficient DFT.

An Efficient Implementation of a DFT

This technique is applicable to DFT's where N

is a prime number, but where N is not prime, a known technique can be used to factorise the transform into prime number transforms, each implementable as described. The implementation is developed using N=5 as an example, although any such prime number could be used. (The case for N=2 is trivial and not appropriate to this method).

The derivation of the efficient transform is simplified when N is prime, since all powers of w appear once and only once in every row of the W matrix in equation (except for row zero). This imparts a high degree of regularity to the subsequent manipulation of this equation.

The terms in the W matrix (i.e. the powers of w in equation (3) can be re-written by noting that

$$w^{p*} = w^{(N-p)} \qquad p\ [1,\ (N-1)/2] \qquad (6)$$

(w^{p*} denotes the complex conjugate of w^{p})

or in particular

$$w^{4} = w^{1*},\ w^{3} = w^{2*} \qquad (6b)$$

When equation (6b) is substituted in equation (3) we get:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} w^0 & w^0 & w^0 & w^0 & w^0 \\ w^0 & w^1 & w^2 & w^{2*} & w^{1*} \\ w^0 & w^2 & w^{1*} & w^1 & w^{2*} \\ w^0 & w^{2*} & w^1 & w^{1*} & w^2 \\ w^0 & w^{1*} & w^{2*} & w^2 & w^1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \qquad (7)$$

Ignoring row 0 of equation (7), all elements in column 4 of the W matrix in equation (7) are the conjugates of elements in the same row in column 1 and likewise for columns 3 and 2 (the leftmost column os column 0). This allows some saving in computation.

The complex multiplications needed to evaluate equation (7) can be taken in pairs (by selecting those involving multiplication by a coefficient and by its conjugate), and these are now examined.

Let the real and imaginary components of a

complex variable be denoted by the upper case representation of that variable, with either 'r' or 'i' appended to indicate real or imaginary parts respectively.

Thus $x_k = Xr_k + jXi_k$ and $c = Cr + jCi$ etc.

Then the sum of a multiplication by a coefficient and its conjugate can be expanded as follows:

$$y = c.x_k + c^*.x_{N-k} \quad (8a)$$

$$= (Cr + jCi).(Xr_k + jXi_k) + (Cr - jCi).(Xr_{N-k} + jXi_{N-k}) \quad (8b)$$

$$= (Cr.(Xr_k + Xr_{N-k}) - Ci.(Xi_k - Xi_{N-k})) + j\,(Cr.(Xi_k + Xi_{N-k}) + Ci.(Xr_k - Xr_{N-k})) \quad (8c)$$

likewise

$$z = c^*.x_k + c.x_{N-k} \quad (9a)$$

$$= (Cr - jCi)\,(Xr_k + jXi_k) + (Cr + jCi)\,(Xr_{N-k} + jXi_{N-k}) \quad (9b)$$

$$= (Cr.(Xr_k + Xr_{N-k}) + Ci.(Xi_k - Xi_{N-k})) + j\,(Cr.(Xi_k + Xi_{n-k}) - Ci.(Xr_k - Xr_{N-k})) \quad (9c)$$

Some new variables can be introduced to present the sum and difference terms in equation (8c) and (9c).

$$u_0 = x_0$$
$$u_k = x_k + x_{N-k}$$
$$v_k = x_k - x_{N-k} \quad k \in [1, (N-1)/2] \quad (10a)$$

Or expressed in terms of real or imaginary components

$$Ur_k = Xr_k + Xr_{N-k}$$
$$Ui_k = Xi_k + Xi_{N-k}$$
$$Vr_k = Xr_k - Xr_{N-k} \quad (10b)$$
$$Vi_k = Xi_k - Xi_{N-k}$$

Using these variables, equations (8), (9) can be re-written as follows :-

$$y = [Cr.Ur_k - Ci.Vi_k] + j\,[Cr.Ui_k + Ci.Vr_k] \quad (11)$$

$$z = [Cr.Ur_k + Ci.Vi_k] + j\,[Cr.Ui_k - Ci.Vr_k] \quad (12)$$

To make for a more concise description, a notation is introduced for the operation expressed in equation (8),(9).

i.e. $y = c \# (x_k, x_{N-k}) = c.x_k + c^*.x_{N-k}$ (13)

and $z = c \# (x_k, x_{N-k}) = c^*.x_k + c.x_{N-k}$ (14)

The '#' operator can be defined by the following expansion:

$$c \# (x_k, x_{N-k}) = [Cr.(Xr_k + Xr_{N-k}) - Ci.(xi_k - Xi_{N-k})] + j\,[Cr.(Xi_k + Xi_{N-k}) + Ci.(Xr_k - Xr_{N-k})] \quad (15)$$

(i.e. as given in equation (8c) etc.)

(The y and z variables are introduced temporarily for equations (8) .. (14), they are used with a different meaning in other equations herein.)

Using the results of equation (8), (9) and the notation of (13) etc. it is possible to rewrite equation (7) as follows :-

$$\begin{matrix} y_0 & w^0 & w^0 & w^0 & & (x_0,0) \\ y_1 & w^0 & w^1 & w^2 & \# & (x_1,x_4) \\ y_2 & w^0 & w^2 & w^{1*} & & (x_2,x_3) \\ y_3 & w^0 & w^{2*} & w^1 & & \\ y_4 & w^0 & w^{1*} & w^{2*} & & \end{matrix} \quad (16)$$

The '#' operator requires four real multiplications (as does complex multiplication), but the evaluation of equation (16) required in total about half the number of multiplications as does equation (7). Thus a significant saving in the number of multiplications has been achieved by using the relations of equation (8), (9).

As a refinement, the expressions obtained when equation (16) is evaluated can be written down so that their terms are ordered by the indices of w^p.

$$\begin{aligned} Y_0 &= t \\ Y_0 &= x_0 + w^1 \# (x_1, x_4) + w^2 \# (x_2, x_3) \\ Y_2 &= x_0 + w^{1*} \# (x_2, x_3) + w^2 \# (x_1, x_4) \\ Y_3 &= x_0 + w^1 \# (x_2, x_3) + w^2 \# (x_1, x_4) \\ Y_4 &= x_0 + w^1 \# (x_1, x_4) + w^2 \# (x_2, x_3) \end{aligned} \quad (17)$$

where $t = \sum_{q=0}^{N-1} x_q$

Finally, for completeness the entire sequence of operations necessary to calculate a 5 point DFT is as follows :-

First form the u, v and t (complex valued) terms

$$\begin{aligned} u_0 &= x_0 \\ u_1 &= x_1 + x_4 \\ v_1 &= x_1 - x_4 \\ u_2 &= x_2 + x_3 \\ v_2 &= x_2 - x_3 \\ t &= x_0 + x_1 + x_2 + x_3 + x_4 \end{aligned} \qquad (18)$$

Let the real and imaginary parts of the powers of w be written as :

$$\begin{aligned} w^1 &= a_1 + jb_1 \\ w^2 &= a_2 + jb_2 \end{aligned}$$

Then form the y terms, expressed as real and imaginary components

$$\begin{aligned} Yr_0 &= Tr_0 \\ Yi_0 &= Ti_0 \\ Yr_1 &= Ur_0 + a_1.Ur_1 - b_1.Vi_1 + a_2.Ur_2 - b_2.Vi_2 \\ Yi_1 &= Ui_0 + a_1.Ui_1 + b_1.Vr_1 + a_2.Ui_2 + b_2.Vr_2 \\ Yr_2 &= Ur_0 + a_1.Ur_2 + b_1.Vi_2 + a_2.Ur_1 - b_2.Vi_1 \\ Yi_2 &= Ui_0 + a_1.Ui_2 - b_1.Vr_2 + a_2.Ui_1 + b_2.Vr_1 \\ Yr_3 &= Ur_0 + a_1.Ur_2 - b_1.Vi_2 + a_2.Ur_1 + b_2.Vi_1 \\ Yi_3 &= Ui_0 + a_1.Ui_2 + b_1.Vr_2 + a_2.Ui_1 - b_2.Vr_1 \\ Yr_4 &= Ur_0 + a_1.Ur_1 + b_1.Vi_1 + a_2.Ur_2 + b_2.Vi_2 \\ Yi_4 &= Ui_0 + a_1.Ui_1 - b_1.Vr_1 + a_2.Ui_2 - b_2.Vr_2 \end{aligned} \qquad (19)$$

The expressions in equation (19) suggest a useful hardware configuration in which a 'Sum of Products Multiplier', such as described in our co-pending application No.8132315 (T.J.M. Rossiter 3) is loaded with the coefficient values formed by real and imaginary components of w^0, w^1, w^2, that is 1, a_1, b_1, a_2, b_2, and then the data is presented to the relevant inputs to calculate the Yr_p and Yi_p terms. Such a configuration is described below.

The advantages of performing a DFT as described are as follows :-

1. The total number of multiplications needed is approximately half that for a normal DFT.

2. Re-ordering the data (as in equations (17), (19)) allows very efficient use of a particular form of Sum of Products Multiplier.

3. Performing all the non-trivial multiplications in a single step (i.e. evaluating each line of equation (19) completely in a Sum of Products Multiplier) produces smaller numerical rounding errors than do some alternative FFT algorithms.

Extending the Efficient DFT to Other Values of N

We have described how an efficient Fourier transform was derived for N points, N being prime. We now extend it to the case when N is not prime. The techniques used are known and are included to show how the ideas can be applied more widely. We do not give a general treatment, but instead a worked derivation showing how the 5 point transform can be extended to a 10 point transform.

Where N is not prime it can be factorised into a set of factors N_0, N_1, ... N_{M-1}.

$$N = N_0 \times n_1 \times \ldots \times N_{M-1} \qquad (20)$$

Thus it is possible to factorise the N point DFT into a number of smaller DFTs, each implementable using the techniques described above. If the factors have no common prime factors they are said to be mutually prime, and the resultant factorised DFT has particularly attractive properties. But even if the factors are not mutually prime, techniques similar to those described below can be used, though the count of the number of multiplications is higher.

As an example : N=10 has factors 2, 5 (which are mutually prime). A 10 point Fourier transform can be realised as five 2 point transforms followed by two 5 point transforms, see Fig. 1. A further example is given by a 12 point DFT, realised as three 4 point and four 3 point transforms, see Fig. 2.

The correct sequencing of the input and output data is crucial to the operation of the factorised DFT,

but since the derivation is complicated, only the results are presented here.

A useful saving in the number of multiplications occurs when the transforms are factorised. The figures for a regular 10 point DFT, a factorised DFT and a factorised DFT in which the 5 point factors use the FFT algorithm described above are as follows :-

Transform	No. of Real Multiplications
10 pt DFT	304
Factorised	128
FFT factors	64

These figures give the number of non-trivial real multiplications i.e. multiplications involving factors other than unity. Thus the FFT algorithm described above needs (in this case) less than $^1/_4$ the number of multiplications involved in a simple DFT.

Hardware Implementation of the Efficient DFT

The logic needed to implement the efficient DFT is shown in outline in Fig. 3. The data is first processed by a "Sum and Difference Processor" (SDP) which can perform calculations such as those of equation (10). The SDP is followed by an "Addressable Buffer Store" (ABS) which collects the data generated by the SDP and routes it to a "Sum of Products Multiplier" (SPM) e.g. as described in our above mentioned Application, which evaluates expressions of the form of those in equation (19). It can evaluate a number of multiplications simultaneously and form their sum in one data word period.

It is possible to design logic specialised for one particular order of DFT (e.g. an SDP and ABS designed specifically for a 10 point transform), but here we use more general-purpose logic which is usable on a limited, though useful, class of transforms. The operations performed by this logic for any particular

DFT are then determined by control signals, i.e. the logic can be electrically programmed to suit a number of related implementations.

We first define certain terms and then describe the operation of the SDP and ABS, plus brief details of their use in typical DFT computations.

Definition of Terms, Data Formats etc.

Abbreviations used are ABS for Addressable Buffer Store, SDP for Sum and Difference Processor, and SPM for Sum of Products Multiplier.

The following terms are used herein :

Part-Word : An individual real data value (which may be represented for example by a two's complement binary number).

Complex Word : An individual complex data value consisting of two Part-Words (representing the real and the imaginary parts).

Block : A collection of Complex Words forming the input or output data of a DFT i.e. there are N Complex Words in the Block of input data for an N point DFT.

Part-Words are composed of M binary bits, M being constant but not specified, serially sequenced such that the least significant bit occurs first followed by bits of increasing significance. Thus a part-word period is M times the fundamental bit period. Two's complement arithmetic is assumed to be used. Complex words are normally sequenced with the real part preceding the complex part, so that a complex word period is twice a part-word period. Finally, blocks consist of a serially-sequenced collection of complex words. The sequencing is specified for a particular implementation, but a block period is N times a complex word period. These formats are shown in Fig. 4.

The fundamental unit of data in general is the complex word, though at times it may be necessary to manipulate separately the real and imaginary part-word.

Likewise the fundamental unit of time is the complex word period, though the logic will need synchronisation at both the part-word rate and bit rate.

Sum and Difference Processor (SDP)

The SDP shown in outline in Fig. 5 has two parallel paths 'a' and 'b' which are treated slightly differently. The blocks in Fig. 5 have the following functions :

The 'Input Gating' 1 can select either input data or the stored results of previous computations. The selected signal passes to the 'Sum and Difference' circuit 2 which has two outputs. The 'a' path output is the sum of the two paths, the 'b' path output is the difference.

There is a complex word period delay 3 after the sum and difference circuit, which is included to allow certain data re-ordering. The box marked 'xj' in the 'b' path indicates multiplication by 'j'. which can be activated or deactivated. Finally the two paths pass to the store 4 where they can be held for further computation. The store can also swap over the paths (as needed in certain computations).

The output from the SDP is taken after the delay stage 3. It includes a circuit 5 to re-order the real and imaginary part words so that calculations of the form of equation (11), (12) can be performed easily by an SPM. The output can also be passed internally to an accumulator 6, whose output is available separately.

The operation of these various components of the SDP can be modified by control signals, though these are not described in detail. The SDP is sufficiently versatile to be used in the computation of a number of different DFT's, three examples of which are given above.

Addressable Buffer Store (ABS) (Fig. 6)

The purpose of the Addressable Buffer Store (ABS) is to collect a block of data and then allow a

selection of the complex words in that block to be routed to a number of output lines. The store is so configured that one part can be collecting data whilst the other part delivers data collected during a previous block. At the start of the next block, the part which was previously collecting data can be used to deliver it, and the other part can in turn collect fresh data.

The ABS, Fig. 6, consists of two main stores numbered 0 and 1 and logic to load the stores and route the contents of selected locations to the ABS outputs. Each store can operate relatively independently and is internally partitioned into four banks. The facility of loading a store whilst reading previously loaded values is realised by using some banks for loading and others for reading.

The loading of both stores is similar. During each complex word period one bank of store 0 and one bank of store 1 may be loaded, the banks being independently selected. When a store is loaded the existing contents (one complex word per location) 'move down' (e.g. the complex word in 3r moves to 2i and is replaced by that in 3i) and the input is loaded into the topmost location (e.g. 3i). When a store is not selected for loading, the contents remain in the same location. Each location holds a complex word.

There is a slight difference between the two stores in the method of reading data. In store 0 one location from one bank is selected and fed to output Y_0. In store 1 six locations, all from the same bank, can be selected simultaneously. The locations and outputs are grouped in three pairs (the "i" and "r" locations being grouped together) and the pair of locations selected for each output is independently controlled. Each "i" output (Yi_1, Yi_2, Yi_3) has an inversion facility, used to effect the sign inversions necessary for complex multiplication. The

locations accessed by the output selectors can be in any of the four banks, the actual bank selected being determined by an additional control (common to both stores).

By using the various store access controls described above, it is possible to use the ABS to route data to a sum of products multiplier and so compute expressions of the form given in equation (19). It is also possible to use the ABS as a "Timeslot Interchanger", an example of such use could be in arranging the data in the special sequence required by a particular DFT.

Examples of DFT's

The implementation of three FFT's using the SDP, ABS and SPM devices is described. It is possible to apply the SDP, ABS and SPM to a wider range of DFT's than given in these examples, and in fact in the trans-multiplexer application, it was a 14 point application made up of seven two-point devices feeding two seven point devices. In some cases this may require a multiplicity of the three basis devices and/or an ABS device with a greater storage capacity.

Example 1. A 5-point DFT

This is the simplest case : a 5 point transform is performed using the techniques referred to above. The SDP evaluates the expressions given in equation (18) and generates the u, v and t complex terms during a block period. They are then stored in the ABS in the following manner :-

u_0, t store 00 (or 01 during alternate blocks)

u_1, v_1, u_2, v_2 store 10 (or 11 during alternate blocks)

The SPM is loaded with the (real and imaginary) coefficient values for w^0, w^1, w^2 and then the u, v and t terms are routed to the SPM to yield the expressions given in equation (19).

Example 2. A 10 point DFT

In this case the 10 point transform is factorised

into 2 point and 5 point transforms (in the manner shown in Fig. 2.1). The SDP evaluates the five 2 point DFT factors and the sum and difference terms in the 5 point factors. The calculations performed by the SDP are given in equations (21), (22).

$x_0 + x_5 \quad z_0' \longrightarrow z_0 \rightarrow u_0$

$x_0 + x_5 \quad z_1' \longrightarrow z_1 \rightarrow u_1$

$x_2 + x_7 \quad z_2' \longrightarrow z_2 + z_8 \rightarrow u_2$

$x_2 - x_7 \quad z_3' \longrightarrow z_2 - z_8 \rightarrow v_2$

$x_8 + x_3 \quad z_8' \longrightarrow z_3 + z_9 \rightarrow u_3$

$x_8 - x_3 \quad z_9' \longrightarrow z_3 - z_9 \rightarrow v_3$

$x_4 + x_9 \quad z_4' \longrightarrow z_4 + z_6 \rightarrow u_4$

$x_4 - x_9 \quad z_5' \longrightarrow z_4 - z_6 \rightarrow v_4$

$x_6 + x_1 \quad z_6' \longrightarrow z_5 + z_7 \rightarrow u_5$

$x_6 - x_1 \quad z_7' \longrightarrow z_5 - z_7 \rightarrow v_5$ (21)

$t_0 = z_0 + z_2 + z_4 + z_6 + z_8$

$t_1 = z_1 + z_3 + z_5 + z_7 + z_9$ (22)

The SDP interleaves calculations for the 2 point transforms with the pre-processing for the 5 point transforms. The data generated by the SDP is stored in the ABS in the following manner :-

u_0, t_0 store 00 (or 02 during alternate blocks)

u_1, t_1 store 01 (or 03 during alternate blocks)

u_2, v_2, u_4, v_4 store 10 (or 12 during alternate blocks)

u_3, v_3, u_5, v_5 store 11 (or 13 during alternate blocks)

The five point transforms can then be evaluated in a similar manner to the first example (using (u_0, u_2, v_2, u_4, v_4) and (u_1, u_3, v_3, u_5, v_5)) in expressions of the form of equation (19).

Example 3. A 12 point DFT

This is factorised into three 4 point and four 3 point transforms, as shown in Figure 2.2. One SDP evaluates the three 4 point DFT factors; the three point DFTs can either be evaluated directly by an SPM

or a second SDP and SPM with fewer inputs. The calculations performed by the first SDP are given in equation (23), and those by the second SDP in equation (24), (25).

$$
\begin{array}{lll}
x_0 + x_6 & s_0 & s_0 + s_2 \rightarrow z_0 \\
x_0 - x_6 & s_1 & s_0 - s_2 \rightarrow z_2 \\
x_3 + x_9 & s_2 & s_1 + js_3 \rightarrow z_1 \\
x_3 - x_9 & s_3 & s_1 - js_3 \rightarrow z_3 \\
\hline
x_4 + x_{10} & s_4 & s_4 + s_6 \rightarrow z_4 \\
x_4 - x_{10} & s_5 & s_4 - s_6 \rightarrow z_6 \\
x_7 + x_1 & s_6 & s_5 + js_7 \rightarrow z_5 \\
x_7 - x_1 & s_7 & s_5 - js_7 \rightarrow z_7 \\
\hline
x_8 + x_2 & s_8 & s_8 + s_{10} \rightarrow z_8 \\
x_8 - x_2 & s_9 & s_8 - s_{10} \rightarrow z_{10} \\
x_{11} + x_5 & s_{10} & s_9 + js_{11} \rightarrow z_9 \\
x_{11} - x_5 & s_{11} & s_9 - js_{11} \rightarrow z_{11}
\end{array}
\qquad (23)
$$

$$
\begin{array}{ll}
z_0 & u_0 \\
z_4 + z_8 & u_4 \\
z_4 - z_8 & v_4 \\
\hline
z_1 & u_1 \\
z_6 + z_9 & u_6 \\
z_5 - z_9 & v_5 \\
\hline
z_2 & u_2 \\
z_6 + z_{10} & u_6 \\
z_6 - z_{10} & v_6 \\
\hline
z_3 & u_3 \\
z_7 + z_{11} & u_7 \\
z_7 - z_{11} & v_7
\end{array}
\qquad (24)
$$

$$
\begin{aligned}
t_0 &= z_0 + z_4 + z_8 \\
t_1 &= z_1 + z_5 + z_9 \\
t_0 &= z_2 + z_6 + z_{10} \\
t_0 &= z_3 + z_7 + z_{11}
\end{aligned}
\qquad (25)
$$

The output of the SDP is stored in the ABS in the following manner :-

u_0, u_1, u_2, u_3	store 00
t_0, t_1, t_2, t_3	store 01
u_4, v_4, u_5, v_5	store 10
u_6, v_6, u_7, v_7	store 11

Finally the SPM computes expressions of the form:-

$$
\begin{aligned}
Yr_0 &= Tr_0 \\
Yi_0 &= Ti_0 \\
Yr_4 &= Ur_0 + a_1.Ur_4 - b_1.Vi_4 \\
Yi_4 &= Ui_0 + a_1.Ui_4 + b_1.Vr_4 \\
Yr_8 &= Ur_0 + a_1.Ur_4 + b_1.Vi_4 \\
Yi_8 &= Ui_0 + a_1.Ui_4 - b_1.Vr_4
\end{aligned}
\qquad (26)
$$

CLAIMS :

1. An electronic circuit for the implementation of a discrete Fourier transform (DFT), which includes input means top which are applied electrical signals representing quantities on which the DFT is to be performed, and a sum and difference processor (SDP) to which those signals are applied and which can perform calculations of the following types on the signals :-

$$u_o = x_o$$
$$u_k = x_k + x_{N-k}$$
$$v_k = x_k - x_{N-k}$$

where x_2 are a sequence of input samples and N is the order of the matrix, x is an input signal and U and V are the results of the computations, characterised in this, that the results generated by the SDP are collected in an addressable buffer store, and that the results as assembled in the addressable buffer store are applied to a sum of products multiplier (SPM) which evaluates expressions in the following format, quoted for an example in which N = 5;

$$Yi_0 = Ti_0$$
$$Yr_1 = Ur_0 + a_1 Ur_1 - b_1 Vi_1 + a_2 Ur_2 - b_2 Vi_2$$
$$Yi_1 = Ui_0 + a_1 Ui_1 + b_1 Vr_1 + a_2 Ui_2 + b_2 Vr_2$$
$$Yr_2 = Ur_0 + a_1 Ur_2 + b_1 vi_2 + a_2 Ur_1 - b_2 Vi_1$$
$$Yi_2 = Ui_0 + a_1 Ui_2 - b_1 Vr_2 + a_2 Ui_1 + b_2 Vr_1$$
$$Yr_3 = Ur_0 + a_1 Ur_2 - b_1 Vi_2 + a_2 Ur_1 + b_2 Vi_1$$
$$Yi_3 = Ui_0 + a_1 Ui_2 + b_1 Vr_2 + a_2 Ui_1 - b_2 Vr_1$$
$$Yr_4 = Ur_0 + a_1 Ur_1 + b_1 Vi_1 + a_2 Ur_2 + b_2 Vi_2$$
$$Yr_4 = Ur_0 + a_1 Ur_1 + b_1 Vi_1 + a_2 Ur_2 + b_2 Vi_2$$
$$Yi_4 = Ui_0 + a_1 Ui_1 - b_1 Vr_1 + a_2 Ui_2 - b_2 Vr_2$$

where Yr and Yi are real and imaginary results respectively of the DFT, where a_1, b_1, a_2, b_2 are the real and imaginary Fourier coefficients, and where

$$a_1 + jb1 = \exp\left(j\frac{2}{5}\right),\ a_2 + jb2 = \exp\left(j\frac{2}{5}\right)$$

$$Tr_0 = Ur_0 + Ur_1 + Ur_2$$
$$Ti_0 = Ui_0 + Ui_1 + Ui_2,$$

the resulting output signals from the SPM being representative of a DFT of the quantities represented by the electrical signals applied to said input means.

2. A circuit arrangement for converting the signals from a number of PCM channels into a frequency division mulitplex (FDM) group, characterised by means to convert the PCM words into their linear representations, connections over which said linear representations are applied to the said input means, and a polyphase filter to which the said output signals are applied.

3. An electronic circuit for the implementation of an inverse Fourier transform (IFT), which uses a circuit as claimed in claim 1, modified in that it operates in the reverse direction, i.e. effecting the conversion from PCM to FDM.

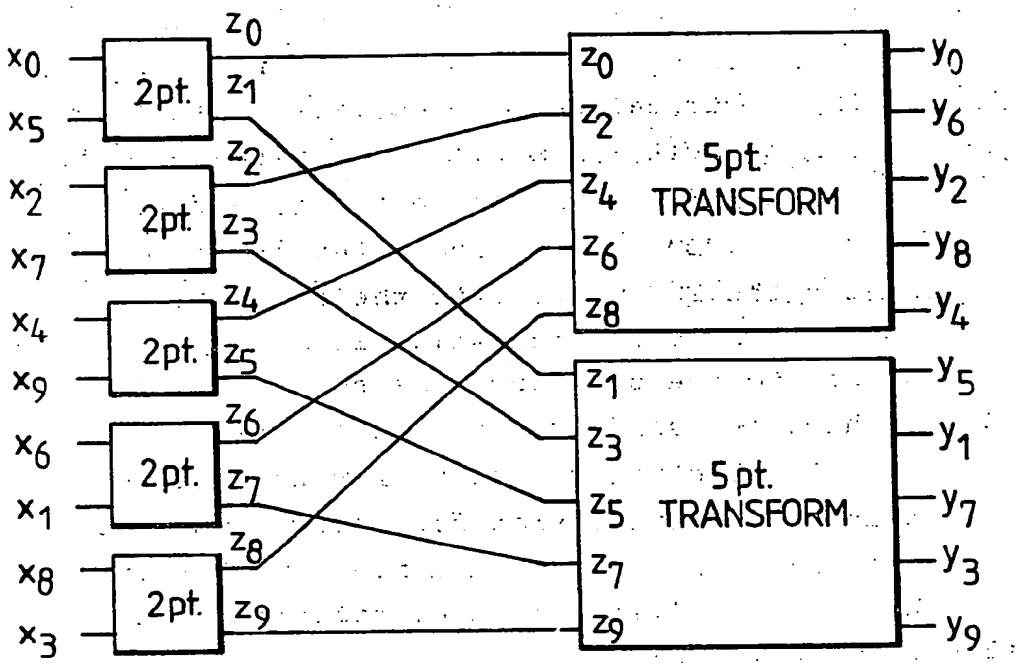

x0
x5
x2
x7
x4
x9
x6
x1
x8
x3
2pt.
2pt.
2pt.
2pt.
2pt.
z0
z1
z2
z3
z4
z5
z6
z7
z8
z9
5pt TRANSFORM
5 pt. TRANSFORM
y0
y6
y2
y8
y4
y5
y1
y7
y3
y9

Fig.1

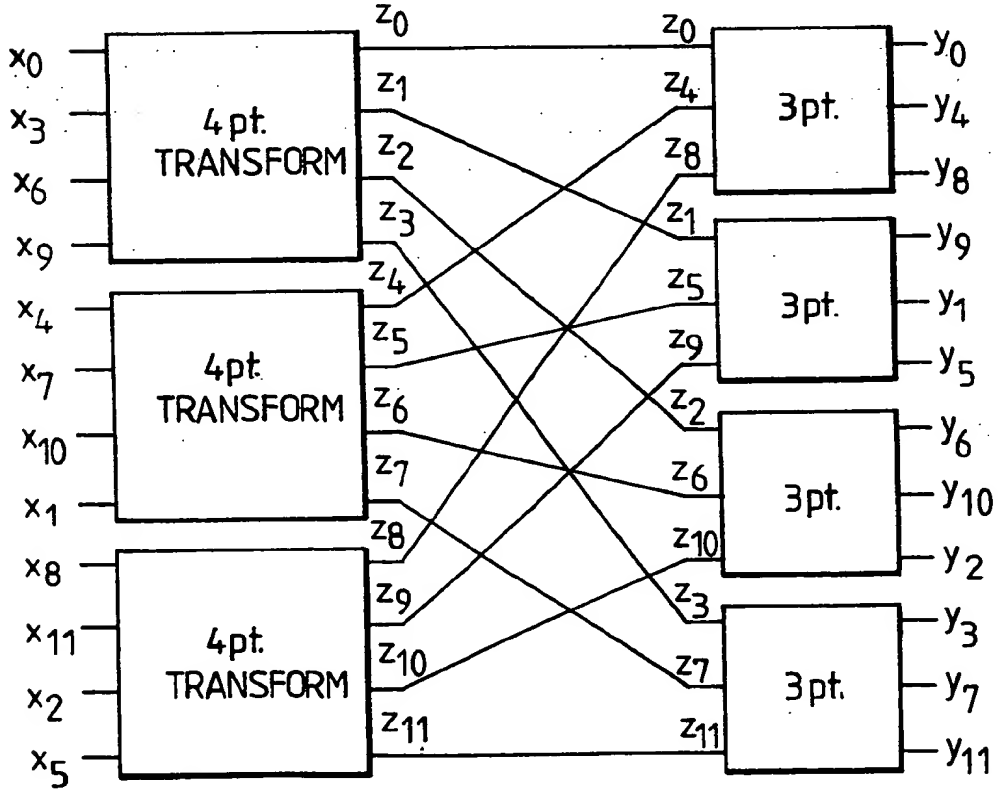

x0
x3
x6
x9
x4
x7
x10
x1
x8
x11
x2
x5
4 pt. TRANSFORM
4pt. TRANSFORM
4pt. TRANSFORM
z0
z1
z2
z3
z4
z5
z6
z7
z8
z9
z10
z11
3pt.
3pt.
3pt.
3pt.
y0
y4
y8
y9
y1
y5
y6
y10
y2
y3
y7
y11

Fig.2

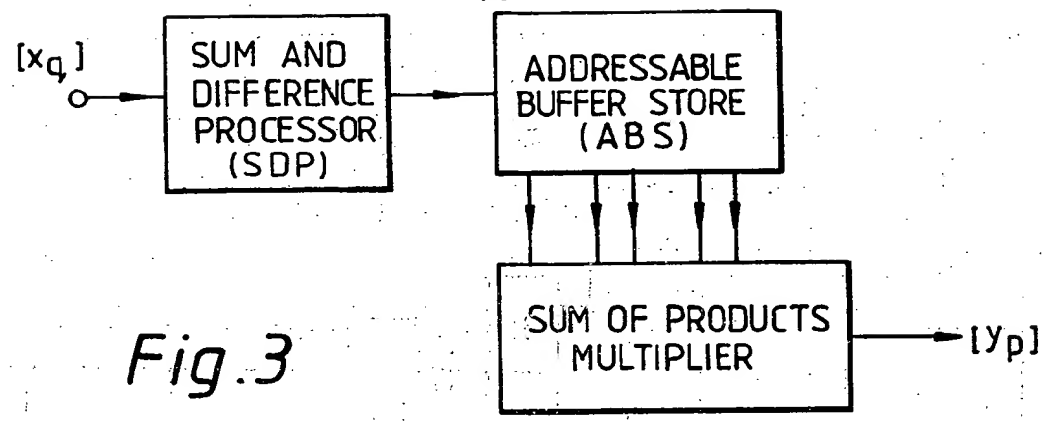


Fig.3

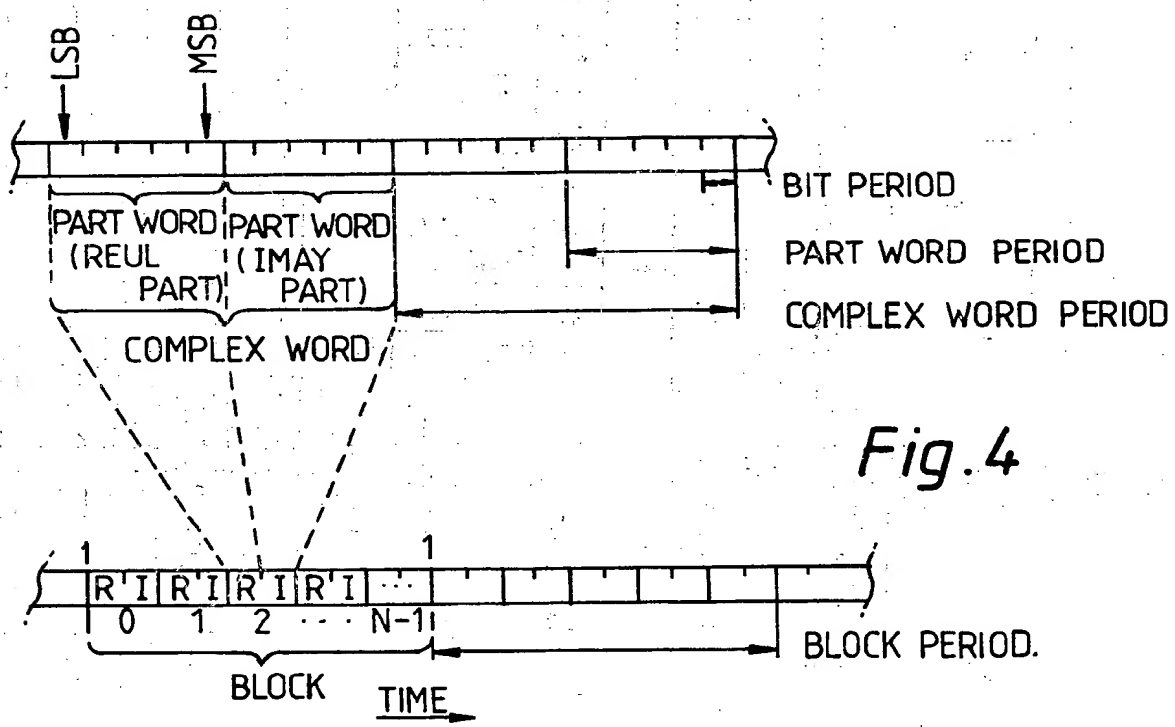


Fig.4

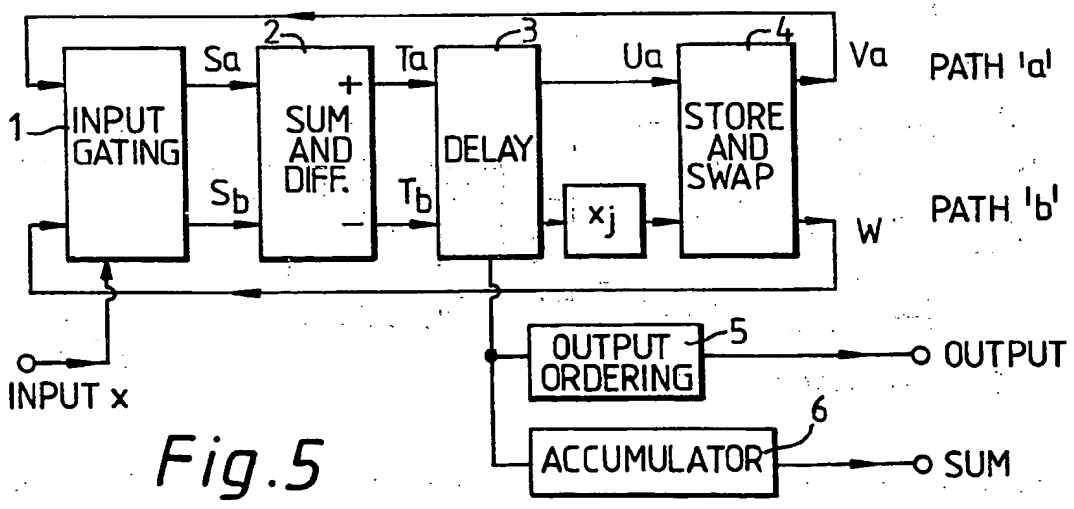


Fig.5

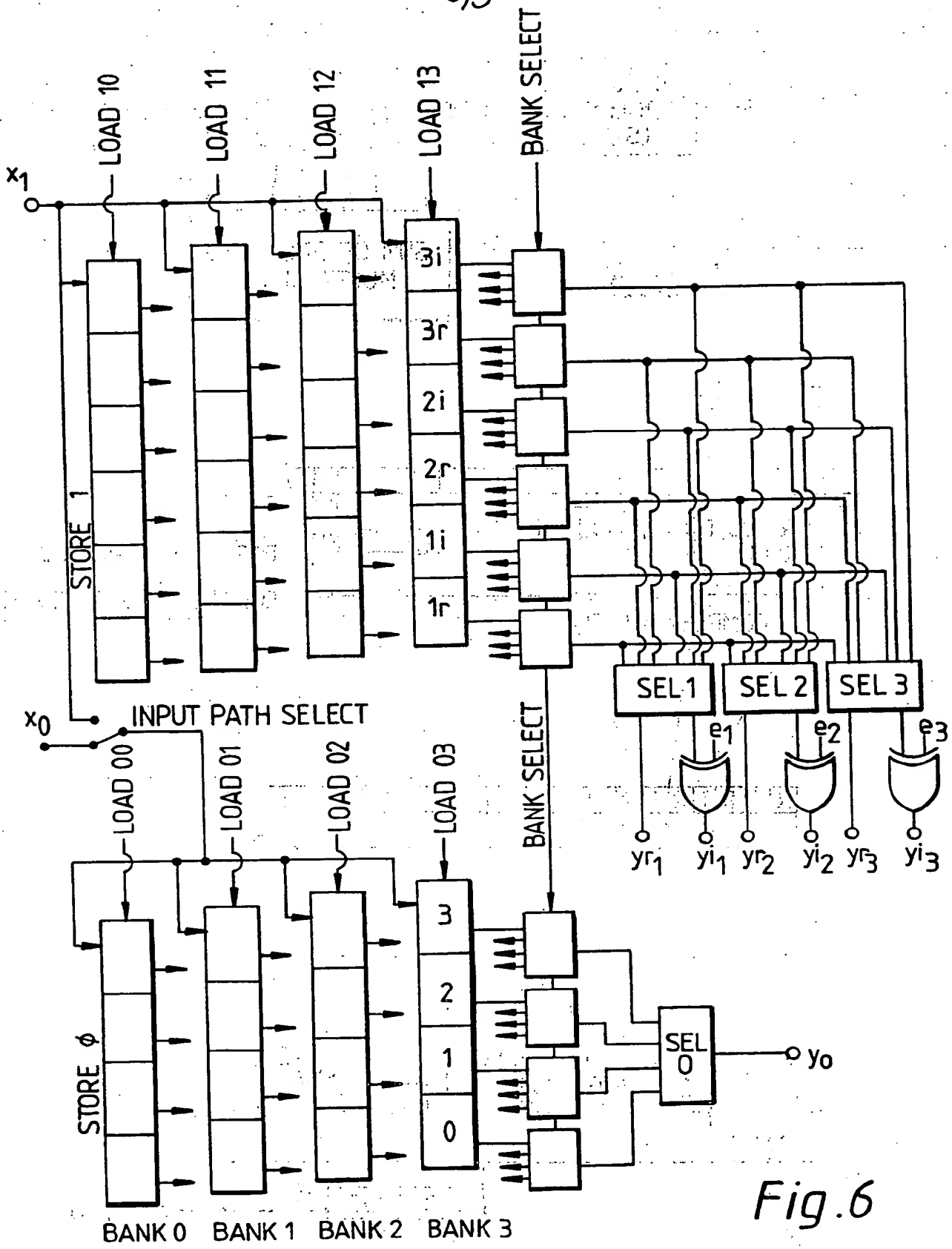

LOAD 10
LOAD 11
LOAD 12
LOAD 13
BANK SELECT
x1
STORE 1
3i
3r
2i
2r
1i
1r
SEL 1
SEL 2
SEL 3
e1
e2
e3
yr1
yi1
yr2
yi2
yr3
yi3
x0
INPUT PATH SELECT
LOAD 00
LOAD 01
LOAD 02
LOAD 03
BANK SELECT
STORE φ
3
2
1
0
SEL 0
yo
BANK 0
BANK 1
BANK 2
BANK 3

Fig.6